Achieving Loop-Free Incremental Routing in Ad Hoc Networks*

Hari Rangarajan J. J. Garcia-Luna-Aceves
Department of Computer Engineering
University of California
Santa Cruz, CA 95064, U.S.A.
Email: {hari, jj}@cse.ucsc.edu

Abstract

We present the Dynamic Incremental Routing (DIR) protocol, which features instantaneous loop-free routing of data packets based on their destination addresses (hop-by-hop routing). Loop-free routes are maintained by using "feasible distances" to order the nodes with respect to a destination. Simulation results show that the performance of DIR is much better than the performance of AODV, DSR and OLSR, which are indicative of the state of the art in routing protocols.

1 Introduction

Several routing protocols have been proposed to date for ad hoc networks. These protocols are either pro-active (routes are maintained to every possible destination in the network) or on-demand (routes are established upon request). Pro-active protocols like the Optimized Link State Routing (OLSR) [1], Source Tree Adaptive Routing (STAR) [5], and Topology Broadcast Reverse Path Forwarding (TBRPF) [7] incur temporary loops, whereas such on-demand routing protocols like the Ad hoc On-demand Distance Vector (AODV) [9] protocol, the Dynamic Source Routing (DSR) [6] protocol, and the Temporally Ordered Routing Algorithm (TORA) [8], ensure loop freedom at every instant.

DSR establishes a loop-free route to a destination by carrying the path traversed in the route request and the reverse path is then used to source route data packets. Although this approach ensures that data packets do not traverse loops, it incurs additional overhead in packet headers. On a link failure, *reliable* error notifications have to be sent to the source, so that a new route can be searched. The DSR draft [6] defines an operation to recover from link failures locally, by re-routing data packets along an alternative source route called packet salvaging. However, this approach results in the formation of loops and requires a mechanism to detect packets flowing in loops.

TORA establishes multiple routes to a destination on-demand. TORA uses link-reversal [2] to recover from link failures by ordering nodes with a combination of route establishment timestamps and a height associated at each node per destination. A route is clamped down at a node during the local recovery phase and all updates need to be reliable for the algorithm to work. The recovery will re-route along a alternative path which was discovered during the initial route equest and it does not take into account availability of better routes due to mobility. TORA's route setup incurs more control overhead than other on-demand routing protocols because of their need for reliable communication updates during route setup.

AODV maintains loop freedom using destination sequence numbers as a routing invariant. Route requests carry the last-known sequence number, which elicits route replies with a equal or higher sequence number. Error updates can be sent unreliably, because a node increases its sequence number for a destination upon the occurrence of a link failure, and invalidates the route. This prevents upstream nodes from replying to any route requests. The Labeled Distance Routing (LDR) [4] protocol improves on the way in which AODV uses sequence numbers by using an additional invariant based on distance. LDR orders nodes by using the last known shortest distance (feasible distance) with respect to a destination, and the destination sequence number is used as a "reset" when no path which satisfies the distance ordering can be obtained.

Loop freedom can be achieved in a routing protocol if stale updates can be sorted out from fresh ones or establishing an ordering along nodes on the path to the destination, and this is usually achieved by using a combination of timestamps or sequence numbers. The problem with using timestamps is the need for a globally synchronized clock at all nodes, whereas sequence numbers would need to be

*This work was supported in part by the US Air Force/OSR under Grant No. F49620-00-1-0330, and by the Baskin Chair of Computer Engineering at UCSC.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
2004		**00-00-2004 to 00-00-2004**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Achieving Loop-Free Incremental Routing in Ad Hoc Networks	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of California at Santa Cruz,Department of Computer Engineering,Santa Cruz,CA,95064**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**6**	

Table 1. Terminology

Notation	Description
d_D^A	The stored distance for destination D at node A.
s_D^A	The successor for destination D at node A.
PS_D^A	The set of neighbors of node A to whom node A has sent RREPs for D.
fd_D^A	The least distance assigned by node A for D since $PS_D^A = \phi$.
c_B^A	The cost of the link from node A to neighbor B.
req,rep	Superscript used for variables in a route request and a route reply.

reset eventually, which requires reliable broadcasts.

This motivates the design for a new routing protocol that performs incremental (i.e, hop-by-hop) routing, uses unreliable error updates, requires no sequence number or timestamps, and no synchronization with neighbor nodes. We present the Dynamic Incremental Routing (DIR) protocol, an on-demand routing protocol for mobile wireless ad hoc networks (MANET). A strict ordering is established using distances to destinations instead of sequence numbers or timestamps, allowing DIR to avoid synchronization with any neighbor. Instantaneous loop freedom is achieved by nodes maintaining their feasible distances (i.e., the smallest distances attained to a given destination), until they can be safely reset (increased) and establishing paths that maintain the feasible distance ordering during route maintenance operations.

Section 2 provides a detailed description of DIR. Section 3 provides an example of DIR's operation. Performance of DIR against two on-demand routing protocols (AODV, DSR) and a pro-active link state protocol (OLSR) is presented in Section 4. Section 5 provides our concluding remarks.

2 Dynamic Incremental Routing Protocol (DIR)

2.1 Achieving Loop freedom

DIR uses route request (RREQ), route reply (RREP) and route error (RERR) messages similar to other on-demand routing protocols. We use the terminology in Table. 1 to describe DIR. DIR relies on the following four rules (called conditions), which apply for a given destination D independently of other destinations. The rules make use of the minimum feasible distance of any of the nodes that relayed or originated a RREQ for destination D (denoted by mfd_D^{req}), and the current feasible distance of the node that transmits a RREP for D (denoted by fd_D^{rep}).

*A*DC: (Accept Distance Condition). When node A receives a RREP from node B for destination D, then Node A sets $s_D^A \leftarrow B$ if ($d_D^{rep} + lc_B^A < d_D^A$) and $fd_D^{rep} \not> fd_D^A$.

*S*DC: (Start Distance Condition). Node I can issue a RREP in response to a RREQ for destination D if I has an active route to D and $d_D^I < mfd_D^{req}$.

*M*DC: (Minimum Distance Condition). If node A relays a RREP for destination D, it sets $fd_D^{rep} = fd_D^A$ and $d_D^{rep} = d_D^A$. Node A relays a RREQ for destination D only if $A \notin path_D^{req}$ and sets $mfd_D^{req} = min(mfd_D^{req}, fd_D^A)$.

*R*DC: (Reset Distance Condition). If node A must change sd_D^A, then it sets $d_D^A \leftarrow \infty$. Node A can set $fd_D^A \leftarrow \infty$ if $d_D^A = \infty$ and $PS_D^A = \phi$.

The key to loop-freedom in DIR is the dissemination of route requests (RREQ) that form a directed acyclic graph (DAG) rooted at the source, and having the route replies (RREP) traverse the reverse paths along such a DAG. RREPs establish a strict ordering of feasible distances when setting up the successor path to the destination. A RREQ traverses a loop-free path and carries the minimum feasible distance of any of its relays (MDC), and only a node with a distance strictly smaller than the minimum feasible distance can generate a RREP (SDC). Together, MDC and SDC ensure that a intermediate node replying has a loop-free path to the destination which does not contain any of the nodes in the traversed path. Because the RREP travels a loop-free path, route establishment does not create loops. However, it is essential that a strict ordering of feasible distances be established along the route traversed by the RREP. Reseting feasible distances without synchronization with upstream neighbors can create loops. A technique for reseting feasible distances is to use diffusing computations [3] which requires nodal synchronization across multiple hops. DIR uses RDC, which defines a safe condition by which nodes can determine when their feasible distance can be reset. When the predecessor set of a node is empty, the node has no upstream nodes with whom it needs to synchronize and can reset its feasible distance.

2.2 Information Stored and exchanged

Node A running DIR maintains the following entries in the routing able: (a) the successor to D (s_D^A), (b) the predecessor set for D (PS_D^A) (c) Maximum lifetime of predecessors PS_D^A ($maxlifePS_D^A$) (d) the current distance (d_D^A), and (e) the feasible distance (fd_D^A). The entry $maxlifePS_D^A$ is the time after which nodes in PS_D^A having A as the next hop towards destination D will invalidate their routes.

A RREQ consists of the tuple $\{dst,\ src,\ rreqid,\ mfd^{req}_{dst},\ path^{req}_{dst}\}$, where src represents the source seeking a route to destination dst. The $rreqid$ is an identifier assigned to a RREQ at the source such that each $(src, rreqid)$ pair is unique. The $path^{req}_{dst}$ consists of a set of tuples specifying the path traversed by the RREQ. The tuple for the i^{th} node in the path traversed is of the form $\{node_i,\ maxfd_i,\ lc_i\}$, where $node_i$ is the node identifier for the i^{th} path entry, $maxfd_i$ is the maximum feasible distance that can be set at $node_i$ on receiving a reply. The field lc_i represents the associated link cost.

The RREP consists of the tuple $\{dst,\ src,\ ttl,\ d^{rep}_{dst},\ fd^{rep}_{dst},\ path\}$. The field src specifies the origin of the RREQ for which a RREP was initiated. The field ttl is the lifetime of the route at the node transmitting the RREP. The current distance towards the destination at the node relaying the route reply is represented by d^{rep}_{dst}, and fd^{rep}_{dst} represents the feasible distance.

A RERR message consists of the tuple $\{orig,\ dests\}$, where $orig$ is the originator of the route error message and $dests$ is the list of destinations which are no longer reachable through $orig$.

2.3 Route Maintenance

2.3.1 Initiating a RREQ

Node A is said to be *active* for the route computation for the destination D when it issues a RREQ that is uniquely identified by the pair (A, ID_A). A node can be active for only a single computation for a destination at any instant of time. The route computation (A, ID_A) terminates when node A receives a RREP that is feasible at node A or the timer for the RREQ expires. The route computation terminates as success and a failure in the two cases respectively. A node A that requires a route for destination D buffers packets if it is active for destination D. Otherwise, it issues a RREQ $\{D,\ A,\ rreqid = ID_A,\ mfd^{req}_{dst},\ path^{req}_{D}\}$ by setting $ID_A \leftarrow RequestCounter + 1$; $rreqid \leftarrow ID_A$; $mfd^{req}_{D} \leftarrow fd^{A}_{D}$; $path \leftarrow (nodeId = A,\ maxfd = fd^{A}_{D},\ 0)$ and $RREQtimer \leftarrow (2.ttl.latency)$, where ttl is the time-to-live of the broadcast flood and latency is the estimated per-hop latency of the network.

If node A receives no RREP for destination D after its timer for (A, ID_A) RREQ expires, itsends a new RREQ with an increased ttl. If after a number of attempts node A does not receive a RREP, a failure is reported to the upper layer. The number of hops that a RREQ can traverse is controlled externally from the RREQ by means of the TTL field of the IP packet in which the RREQ is encapsulated.

2.3.2 Relaying RREQs

A node relaying a RREQ originated by another node is said to be *engaged* in the RREQ. A node maybe engaged in multiple RREQs for the same destination, and a node that is engaged in a route computation maintains no explicit state.

When node B receives RREQ $\{D,\ A,\ reqid = ID_A,\ mfd^{req}_{dst},\ path^{req}_{D}\}$, it first determines its own status for (A, ID_A). If B is active (i.e., $B = A$) or engaged (i.e., B is listed in $path^{req}_{D}$) in the computation (A, ID_A), it silently drops the RREQ. Otherwise, node B is said to be *passive*. If this is the case and SDC is satisfied (i.e., $d^{B}_{D} < mfd^{req}_{D}$), then node B issues a RREP (Section 2.3.3). Else, if SDC is not satisfied, node B relays the RREQ with $mfd^{req}_{D} \leftarrow min\{fd^{B}_{D},\ mfd^{req}_{D}\}$ and $path^{req}_{D} \leftarrow path^{req}_{D} \oplus (nodeId = B,\ maxfd = mfd^{req}_{D},\ lc^{B}_{A})$.

2.3.3 Initiating and Processing RREPs

When node I processes RREQ $\{D,\ A,\ reqid = ID_A,\ mfd^{req}_{D},\ path^{req}_{D}\}$ and SDC is satisfied (i.e., $d^{I}_{D} < mfd^{req}_{D}$), it issues a RREP $\{D,\ src,\ d^{rep}_{D}, fd^{rep}_{D},\ ttl,\ path^{rep}_{D}\}$ where $fd^{rep}_{D} \leftarrow fd^{I}_{D}$ and $d^{rep}_{D} \leftarrow d^{A}_{D}$. The destination or the intermediate node initiating the RREP sets $path^{rep}_{D}$ to the reverse path $path^{req}_{D}$ traversed by the RREQ.

If node A receives RREP $\{D,\ src = S,\ fd^{rep}_{D},\ ttl,\ path^{rep}_{D}\}$, it determines if it is the source S of the RREQ that caused the RREP. If so, it proceeds as Section 2.3.4 describes. If $A \neq S$, then it updates its routing table as Section 2.3.4 states, and forwards the RREP along $path^{rep}_{D}$ setting $fd^{rep}_{D} \leftarrow fd^{A}_{D}$ and $d^{rep}_{D} \leftarrow d^{A}_{D}$ if node A updated its routing table after processing the RREP and $fd^{A}_{D} \leq maxfd^{rep_{path.A}}_{D}$. Otherwise, the RREP is silently dropped. Node A also adds the next hop of the RREP, B, to PS^{A}_{D}.

2.3.4 Adding and Updating Routes

By definition, if node A has no routing-table entry for destination D, its distance and feasible distance for D are assumed to have bash: a: command not found d^{rep}_{D},fd^{rep}_{D},ttl, $path^{rep}_{D}\}$ from neighbor B, it carries out the following steps: If $(d^{rep}_{D} + lc^{A}_{B} < d^{A}_{D} \ \wedge \ fd^{A}_{rep} \not> fd^{A}_{D})$, then node A sets $s^{A}_{D} \leftarrow B$, $d^{A}_{D} \leftarrow d^{rep}_{D} + lc^{A}_{B}$, and $fd^{A}_{D} \leftarrow min\{fd^{A}_{D}, maxfd^{rep_{path.A}}_{D}, d^{A}_{D}\}$.

2.3.5 Route Failures

Node A sets $s^{A}_{D} \leftarrow nil$, $d^{A}_{D} \leftarrow \infty$ if no data packets have been forwarded using this route entry for $active\_route\_time$ seconds. Nodes should update lifetimes as discussed in Section 2.3.6.

Node A carries out the following steps if its predecessor set (PS^{A}_{D}) is not empty and either node A receives a link-level notification that its link to s^{A}_{D} has failed, or node A

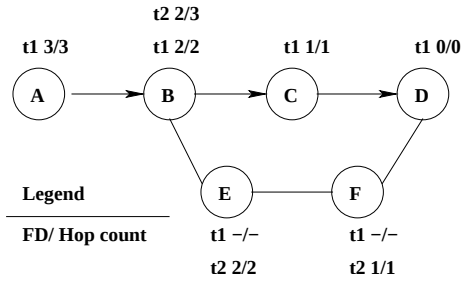

Figure 1. Illustration of DIR

receives a RERR from s_D^A: Node A broadcasts a RERR, and it sends a RREQ for D (Section 2.3.1) if node A is an active source of data packets for D or is performing a localized route repair operation.

2.3.6 Reseting feasible distances

A node can safely set its feasible distance to ∞ when it has an invalid route and the set of predecessors $PS_D^A = \phi$. One approach to ensure the predecessor set is empty is notify each predecessor with a reliable route error update. However, this is expensive in a wireless network with a contention-based MAC protocol.

A node broadcasts a route error (RERR) unreliably to notify its predecessors. The unreliable transmissions of RERRs, together with queueing delays for data packets, can cause route lifetimes to be refreshed although no data packets are delivered. This can result in a node invalidating its route, while the predecessors still have a valid route (because the broadcast route error was lost). A node increasing its feasible distance by incorrectly judging the predecessors set as empty can lead to the formation of loops.

Accordingly, route and predecessor lifetimes are updated in DIR using the following rules: Node A having an active route to destination D with next hop B updates the route lifetime (ttl_D^A) only if it receives an acknowledgement from the lower layer for the delivery of the data packet to B. Node A sets $maxlifePS_D^A$ to expire $active\_route\_timeout$ from current time, when it receives a data packet from any predecessor in PS_D^A.

Following these rules when updating the lifetimes will ensure that the routes expire at a node and its predecessors at the same instant. If $maxlifePS_D^A$ expires without any data packets from PS_D^A, then $PS_D^A \leftarrow \phi$ and if $d_D^A = \infty$ then node A sets $fd_D^A \leftarrow \infty$.

3 DIR Example

Figure 1 shows a directed acyclic successor graph for destination D for a six-node network. Nodes A and B have active flows for destination D. The feasible/measured distances are shown at time t_1 after the initial route setup for destination D. At time $t > t_1$, link BC fails and node B broadcasts a route error notifying A that D is unreachable. However, A does not receive the route error.

Node B initiates a new RREQ with $mfd_D^{req} = 2$ because it has $fd_D^B = 2$ and and cannot set it to ∞ until $PS_D^B = \phi$. Nodes E and F do not possess any knowledge of destination D and cannot satisfy SDC. They relay the RREQ with $maxfd_D^{req_{path.F}} = 2$ and $maxfd_D^{req_{path.E}} = 2$, respectively. The RREQ reaches the destination D because no node satisfies SDC. At the destination D, a RREP is initiated with $d_D^{rep}, fd_D^{rep} \leftarrow 0$, which follows the reverse path. Node F updates its hop count $d_D^F \leftarrow 1$, sets $fd_D^F \leftarrow 1$, and relays the reply to node E. Node E updates distance $d_D^E \leftarrow 2$, sets $fd_D^E \leftarrow 2$. Node B processes the reply and updates its distance $d_D^B \leftarrow 3$, and sets its feasible distance $fd_D^B \leftarrow 2$ (as min($fd_D^B = 2, maxfd_D^{rep_{path.B}} = 2, d_D^B = 3$)). Notice that the feasible distances are monotonically non-increasing towards the destination D along a path.

If A does not receive a RERR for destination D after the new route is setup at B, node A can still forward data packets through B for destination D, and with the ordering maintained no loops would form. On the other hand, if A had received the RERR, it would have issued a new RREQ (A, ID_A) to setup a route to the destination.

4 Performance

We present results for DIR over varying loads and mobility. The protocols used for comparison are two on demand protocols DSR and AODV, which reflect the state of the art in on-demand routing, and OLSR which is a pro-active link state protocol. Simulations are run in Qualnet. The parameters are set as in [10]. We use an optimization in DIR, sending directed route requests (DRREQs) which are calculated from the collected path information carried in control messages by using a path selection algorithm (i.e., Dijkstra). The DRREQs contain a source route along which they are routed. If the DRREQ path is not valid then a normal RREQ is flooded on the next attempt after the request timer expires. Otherwise intermediate nodes satisfying SDC or the destination can generate a RREP if they receive a DRREQ.

Simulations are performed on two scenarios, (i) 50-node network with terrain dimensions of 1500m x 300m, (ii) 100-node network with terrain dimensions of 2200m x 600m. Traffic loads are CBR sources with a data packet size of 512 bytes at 4 packets per second. Load is varied by using 10 flows and 30 flows. The MAC layer used is 802.11 with a transmission range of 275m and throughput 2 Mbps. The simulation is run for 900 seconds. Node velocity is set between 1 m/s and 20 m/s. Flows have a mean length of 100 seconds, distributed exponentially. Each combination

(number of nodes, traffic flows, scenario , routing protocol and pause time) is repeated for 9 trials using different random seeds.

We present four metrics. Delivery ratio is the ratio of the packets delivered per client/server CBR flow. Latency is the end to end delay measured for the data packets reaching the server from the client. The *network load* is the total number of control packets (RREQ, RREP, RERR, Hello, TC etc) divided by the received data packets. *Data hops* is the number of hops traversed by each data packet (including initiating and forwarding) divided by the total received packets in the network. This metric takes into account packets dropped due to forwarding along incorrect paths. A higher data hops metric indicates more number of hops data packets traverse without reaching the destination or the in-optimality of the routes. Table 2 and 3 summarizes the results of the different metrics by averaging over all pause times for the 50 and 100 node networks. The columns show the mean value and 95% confidence interval.

Figs 2, 3, 4, and 5 show the delivery ratio. Confidence intervals(95%) are shown with vertical bars in the graphs. DIR has a very consistent performance across all scenarios and outperforms other protocols in most cases. The exception is at high flows, low mobility scenarios where OLSR and DSR seem to do better. In the low-load 10-flow scenarios, DIR has the highest data delivery although in the 50-node scenario AODV's delivery is statistically equivalent(confidence intervals overlap). The next best protocol in terms of data delivery is DSR, followed by OLSR. The lowest latency is OLSR at .0129 $\pm$ .00 followed by DIR at .0159 $\pm$ .00. Significant control overhead is incurred by OLSR and DSR, whereas both AODV and DSR are statistically equal. In the high-load 50-node 30-flow scenario, DIR and OLSR exhibit statistically equal packet delivery. Both DIR and AODV seem to suffer significantly in these scenarios. The low mobility scenarios tend to be over-bound with network partitions and on-demand protocols would attempt to find a route to destinations that are not reachable. This also accounts for the increased network load of these two protocols in this configuration. In practise, an external measure needs to be imposed to avoid excessive attempts to find unreachable destinations. DIR dominates the low-load, 100-node, 10-flow scenarios with the highest packet delivery at .9917 $\pm$ .00, the lowest latency 0.0316 $\pm$ .00 and lowest network load at 0.7598 $\pm$ .09. The next best performance is exhibited by AODV followed by DSR and OLSR. The same trend follows in the high load 100-node 30-flow scenario with DIR dominating, although incurring a bit more control overhead.

All routing protocols forward a comparable number of data packets as indicated by the data-hops metric. The high packet delivery ratio of DIR across all scenarios indicates the correctness of the routes used for forwarding.

5 Conclusion

We have presented the Dynamic Incremental Routing (DIR) protocol for mobile wireless ad hoc networks. DIR achieves instantaneous loop freedom by maintaining a strict ordering of feasible distances towards the destination. DIR avoids using timestamps/sequence numbers as an invariant for loop freedom, eliminating the need for globally synchronized clocks or the necessity for a sequence number reset. The results of simulation experiments comparing DIR with DSR, AODV and OLSR using Qualnet show that DIR achieves better performance than the other protocols while ensuring loop-free incremental routing.

References

[1] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol," Request for Comments 3626, October 2003.

[2] E. M. Gafni and D. P. Bertsekas, "Distributed Algorithms for Generating Loop-Free Routes in Networks with Frequently Changing Topology," *IEEE Trans. Comm.*, COM-29(1):11–18, Jan. 1981.

[3] J. J. Garcia-Lunes-Aceves, "Loop-Free Routing Using Diffusing Computations," *Proc. IEEE/ACM Trans. Networking*, 1(1):130–41, Feb. 1993.

[4] J. J. Garcia-Luna-Aceves, M. Mosko and C. Perkins, "A New Approach to On-Demand Loop-Free Routing in Ad Hoc Networks," *Proc. PODC 2003*, pp. 53-62, Boston, Massachusetts, July 13–16, 2003.

[5] J. J. Garcia-Luna-Aceves and M. Spohn, "Source-Tree Routing in Wireless Networks," *Proc. IEEE ICNP'99*, pp. 273–82, Toronto, Canada, October 31–November 3, 1999.

[6] D. Johnson et al, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)," IETF Internet draft, draft-ietf-manet-dsr-09.txt, April 2003.

[7] R. Ogier et al., "Topology Dissemination Based on Reverse-Path Forwarding (TBRPF),"Request for Comments 3684, February 2004.

[8] V. D. Park and M. S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *IEEE INFOCOM*, pp. 1405–13 vol.3, Apr. 1997.

[9] C. Perkins et al., "Ad hoc On-Demand Distance Vector (AODV) Routing," Request for Comments 3561, July 2003.

[10] C. Perkins et al. "Performance Comparison of Two On-demand Routing Protocols for Ad Hoc Networks," *IEEE Personal Communications*, 8(1):16 – 28, Feb 2001.

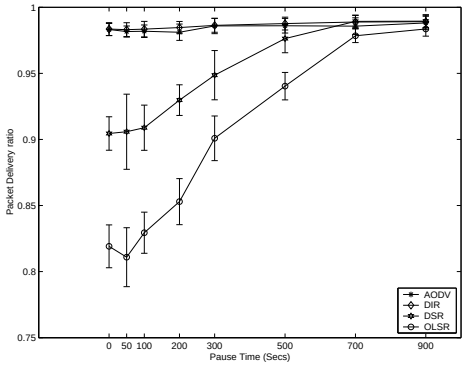

Figure 2. Delivery 50-nodes, 10-flow, 40pps

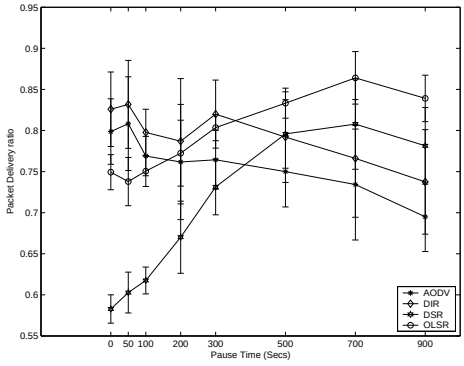

Figure 3. Delivery 50-nodes, 30-flow, 120pps

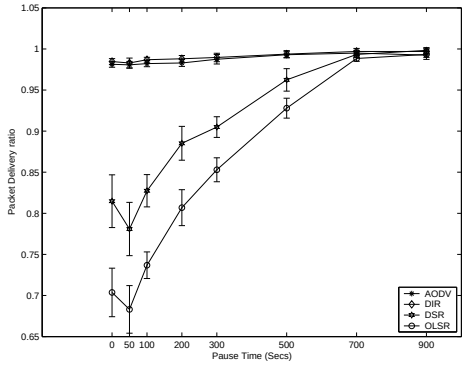

Figure 4. Delivery 100-nodes, 10-flow, 40pps

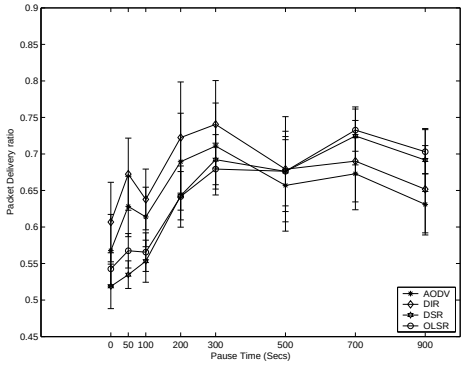

Figure 5. Delivery 100-nodes, 30-flow, 120pps

Table 2. Performance average over all pause times for 50 nodes network

protocol	flows	delivery ratio	latency (sec)	net load	Data Hops
DIR	10	0.9963±0.0006	0.0159±0.0009	0.2237±0.0220	2.6748±0.0723
DSR	10	0.9441±0.0096	0.0323±0.0120	1.7023±0.2986	1.6974±0.0732
OLSR	10	0.8894±0.0163	0.0129±0.0005	2.5773±0.1138	2.5199±0.0705
AODV	10	0.9946±0.0010	0.0170±0.0010	0.2489±0.0248	2.5988±0.0699
DIR	30	0.8147±0.0203	0.7088±0.1261	2.4349±0.3294	2.8901±0.0980
DSR	30	0.6987±0.0236	4.1535±0.4357	1.5269±0.1919	1.9703±0.0851
OLSR	30	0.7938±0.0137	0.8202±0.1138	0.9666±0.0373	2.5845±0.0584
AODV	30	0.7794±0.0186	0.8818±0.1177	3.7867±0.3613	2.9148±0.1043

Table 3. DIR: Performance average over all pause times for 100 nodes network

protocol	flows	delivery ratio	latency (sec)	net load	Data Hops
DIR	10	0.9917±0.0015	0.0316±0.0020	0.7598±0.0975	3.9550±0.1193
DSR	10	0.8959±0.0194	0.0895±0.0430	6.8503±1.1625	3.3228±0.1446
OLSR	10	0.8367±0.0278	0.0224±0.0008	15.1404±0.8074	3.6827±0.1116
AODV	10	0.9886±0.0018	0.0408±0.0042	0.8679±0.1009	3.8248±0.1156
DIR	30	0.6781±0.0222	0.9355±0.0968	8.0876±0.8231	4.5671±0.1640
DSR	30	0.6293±0.0216	4.8421±0.3578	5.5590±0.6150	4.0882±0.1566
OLSR	30	0.6386±0.0191	3.1952±0.3234	6.6386±0.3159	3.9999±0.1216
AODV	30	0.6492±0.0202	1.2072±0.1636	19.1426±7.6063	4.5460±0.1617